Contents

1 Forelesning 1

1.1 Hva er en Algoritme?

En algoritme er en plan for å løse et problem, ikke helt det samme som en implementasjon. Den tar en input og fir en output (metode i *Java*).

Dagens Problem: Gitt en liste med brukernavn, sjekk at ingen brukernavn er like.

1.2 Forelesninger

Forelesningene blir tatt opp og gjort tilgjengelig på Mitt UiB, og kode blir lagt ut på git.

1.3 Pensum

Algorithms by Robert Sedgewick 4^{th} edition. Kapittel 1-4 er pensum.

1. Fundamentale Algoritmer og datastrukturer
2. Sortering
3. Søking
4. Grafer

1.4 Beregne summen fra 1 til n

Følgende formel brukes for å regne summen fra 1 til n.

$$1 + 2 + 3 + \cdots + n = \sum_{i}^{n} \frac{n \cdot (n+1)}{2}$$

Kjøretiden til implementasjonen kan estimeres til

$$\frac{n \cdot (n+1)}{2} \cdot 20 \approx 10n^2$$

$$\begin{aligned} 10 \cdot n^2 &= 10^9 \\ &\approx 10000 \end{aligned}$$

Kjøretid på et tilsvarende program som bruker `Collections.sort()` vil ha en kjøretid på ca.

$$10 \cdot n \log n + 10 \cdot n = 10^9$$

Når n blir stor, er alltid $n \log n$ raskere enn n^2.

1.5 Hvordan forenkle beregninger?

Det er tidkrevende å kjøre programmet og ta tiden. Det er vanskelig å vite nøyaktig hvor mange operasjoner det er, derfor velger vi å beholde største ledd, og vi ser bort ifra konstanter. Som eksempel kan følgende formel forenkles.

$$\frac{n \cdot (n+1)}{2} \cdot 20 =\sim 10n^2 = O\left(n^2\right)$$

2 Forelesning 2

2.1 Hva er en datastruktur

En algoritme tar input, gjør beregninger, og gir output. Datastrukturer lagrer data og har flere forskjellige oppgaver som kan utføres på disse dataene.

> **Example. GPS Navigasjonsenhet**
>
> - Har veinettverk
> - Kan be om korteste vei fra start til mål
>
> Er dette en algoritme eller en datastruktur? ◇

`Collection <E> interface` er en samling med objekter av typen `E`. Viktige metoder er

- `size()`
- `contains(Object O)`
- `add (E e)`
- `Remove(Object obj)`
- `toArray()`

`List` er en utvidelse av `Collection` med litt flere metoder. Elementene i en liste har en indeks, og noen viktige metoder i `List` er

- `indexOf(Object obj)`
- `get(int index)`
- `set(int index, E e)`

2.2 Array List

I `ArrayList` brukes en array av typen `Object[]`. Bare en del av arrayen har data. Når dette arrayet blir fullt må vi lage et større array. Dette betyr at det er lett å få `IndexOutOfBoundsException`. For eksempel kan man lage et nytt array av dobbel størrelse, hvor alle elementer fra forrige array kopieres over til det nye arrayet.

2.3 Linked List

En Linked List er en liste hvor hvert element i listen lagres i et eget node object. Hver node vet kun neste og forrige node. Listen vet kun første og siste node. For å finne node i, så starter vi på første node og "hopper" i ganger.

En linked list som kun vet neste element kalles en *Single Linked List*, og en linked list som vet både neste og forrige element kalles en *Double Linked List*.

2.4 Hvordan finne kjøretid på metoder?

Man må vite hvordan `ArrayList` og `LinkedList` er implementert. Ved å forstå hva `ArrayList` og `LinkedList` gjør, er det lett å forstå hva kjøretiden er. Ukesoppgaven er og implementere enkle versjoner av disse listene.

2.5 Kø og Stabel

Kø	Stabel
FIFO	FILO / LIFO
Legger til sist	Legger til sist
Fjerner først	Fjerner sist

2.5.1 Metoder i Queue og Stack

List	Queue	Stack
add(E e)	offer(E e)	push(E e)
remove(int i)	poll()	pop()
get(int i)	peek()	peek()

2.6 Set

Set er en Collection der hvert element kun kan være der en gang. Elementene har ikke en ebstemt ordning. Set har heller ikke `indexOf(element)` metoden.

	HashSet	TreeSet
`add()`	$O(1)^{\star}$	$O(\log n)$
`remove()`	$O(1)$	$O(\log n)$
`contains(obj)`	$O(1)$	$O(\log n)$

3 Forelesning 3

3.1 Big O Kjøretid

Variablen n er vanligvis definert som størrelse på input, men den kan defineres til hva som helst. Når vi beregner kjøretid er $f(n)$ max antall opersjoner PCen må gjøre når et program kjøres på en input av størrelse n. Det vil si, velg den verste input av størrelse n. Teoretisk er vi interessert i

$$\sum_{n=0}^{\infty} f(n)$$

Mens praktisk er vi interessert i når $f(n)$ er 10^9 til 10^{12}. Vi har at

$$3 + 7 \leq 2 \cdot 7$$
$$124 + 98 \leq 2 \cdot 124$$

Dette gir oss følgende formel

$$a + b \leq 2 \cdot max(a, b)$$

Som resulterer i

$$O(1) + O(1) = O(1)$$
$$O(1) + O(n) = O(n)$$
$$O(n) + O\left(n^2\right) = O\left(n^2\right)$$

Big O beskriver et sett med funksjoner. $n^2 + 3n$ og $7n^2$ er i $O\left(n^2\right)$. Alle funksjoner som er mindre er også med, dvs at $3n$ og $7n \log(n)$ er i $O\left(n^2\right)$.

Definisjon 1. En funksjon $f(n)$ er i $O(g(n))$ dersom det finnes konstanter c og N slik at $f(n) < c \cdot g(n)$ for alle $n > N$, eller, skrevet rent matematisk

$$f(n) \in O(g(n)) \Rightarrow f(n) < c \cdot g(n) \forall\ n > N$$

3.2 Doubling Rate

Hva skjer med kjøretid når input blir dobbelt så stor?

- $O(1)$ - Ingenting, ikke avhengig av n.
- $O(n)$ - $c \cdot n \to c \cdot 2n$, dobbelt så lang tid.
- $O\left(n^2\right)$ - $c \cdot n^2 \to c \cdot (2n)^2$, fire ganger så stort.
- $O\left(n^3\right)$ - $c \cdot n^3 \to c\,(2n)^3$, 8 ganger så stort.